MCGINN & GIBB, PLLC
A PROFESSIONAL LIMITED LIABILITY COMPANY
PATENTS, TRADEMARKS, COPYRIGHTS, AND INTELLECTUAL PROPERTY LAW
8321 OLD COURTHOUSE RD, SUITE 200
VIENNA, VIRGINIA 22182-3817
TELEPHONE (703) 761-4100
FACSIMILE (703) 761-2375

PO9104 09857650

APPLICATION FOR UNITED STATES LETTERS PATENT

APPLICANT: **Mahmoud Meribout**

FOR: **COMPILING METHOD, SYNTHESIZING SYSTEM AND RECORDING MEDIUM**

DOCKET NO.: **19844-01**

明細書

発明の名称　コンパイル方法および合成装置ならびに記録媒体

発明の背景

本発明は、計算機支援用設計（ＣＡＤ）に関し、特には、高級記述言語によるハードウェアモデルの記述が可能な、コンパイル方法および合成装置に関する。さらには、そのようなコンパイル方法を実現するプログラムを記録した記録媒体に関する。

また、本発明は、特定用途向け集積回路（ＡＳＩＣ：Application Specific Integrated Circuit）、現場でプログラム可能なゲートアレイ（ＦＰＧＡ：Field Programmable Gate Array）、および動的再構成可能な論理（ＤＲＬ：Dynamic Reconfigurable Logic）を含む種々の超大規模集積回路（ＶＬＳＩ：Very Large Scale Integration）技術に関する。

高水準回路記述によるハードウェアを合成する装置が一般に知られている。この種の合成装置は、高品質の結果を提供するとともに、設計に際してユーザーに周知の高級言語による記述を提供し、それによって構造的な複雑さからユーザーを解放する。このような合成装置に用いられるコンパイラーは、種々のコストを効率的に用いながら、周知のスケジューリング、アロケーションを実行することにより高いスループットのハードウェアを達成するという利点を有する。

超大規模集積回路（ＶＬＳＩ）の設計では、ゲートをどのようにして相互に接続するかの仕様を伴った、例えばＡＮＤ、ＯＲ、ＮＯＴ、ＦＬＩＰ－ＦＬＯＰなどの２進機能を実行するゲートの集合体が用いられる。そして、設計を適切な技術で作製に適した形に変換するのにレイアウト・ツールが用いられる。このような設計では、「スケマチック・キャプチャ（Schematic Capture）」として知られている既知の手法が使用される。この設計手法によれば、ユーザーは、図形ソフトウエア・ツールを用いて、ライブラリから論理ゲートまたはゲートの集合体を取り出して配置し、コンピュータマウスを用いて配線を「描く」ことにより、それらゲートを相互に接続することができる。その後で、例えばゲートを除去或は単純化することによって、回路全体の機能を変更することなく得られた回路を最適化し、その最適化された回路をレイアウト用および作製用に提示することができる。

しかし、上記の設計手法では、設計者は、全て或は殆ど全てのゲート或はゲートの集合体についての論理とタイミングを考慮しなければならい。そのため、この手法を大規模な設計に使用することは困難であり、また、使用した場合にはエラーを生じやすい。

別の設計技術として、設計者がＬＳＩ回路の記述をハードウェア記述言語（ＨＤＬ）で書くものがある。このＨＤＬにおける記述は、最終設計におけるゲートに対応しており、その入力ソース・コードは最終設計における論理的複雑さと比較して比較的短い。従って、設計者に対する設計の論理的複雑さが軽減される。このようなＨＤＬとしては、IEEE Standard VHDL Language Reference Manual, IEEE Std 1076-1993, IEEE, New York, 1993に開示されたＨＤＬ、およびD.E.Thomas and P.R.MoorbyによりThe Verilog Hardware Description Language, Kluwer Academic 1995に開示されたＶｅｒｉｌｏｇがある。このような言語をS.CarlsonによりIntroduction to HDL-Based Design Using VHDL, Synops Inc., CA, 1991（以下、文献１と称す。）に開示されたような適切な合成ツールとともに用いることにより、設計を回路に変換する。

上述したＨＤＬを用いた合成技術を用いて新たなＶＬＳＩ回路を設計する場合には、以下のような問題を考慮する必要がある。

第１の問題は、シミュレーション時間が長いことである。この問題の解決策は、ディスクまたはランダム・アクセス・メモリ（ＲＡＭ）に保存されている回路に対して、ベクトルとして知られている入力設定を用いたテストをコンパイルし実行するために、標準コンパイラを備えるＣ．Ａ．ワ クステーションとして知られるものが用いられるような、適切な高級プログラミング言語で、ソフトウエア技術者が回路をとらえられるようにすることである。そして、次ステップで、ハードウェア技術者が、前述の参考文献１に開示されている「ＶＨＤＬ Register Transfer Level （ＲＴＬ）」のようなハードウェア合成およびシミュレーションのために、より適切な言語でＣコードを書き換えるようにする。しかし、その場合、ＣバージョンとＨＤＬバージョンとの間は直接関連付けられていないので、ＨＤＬの記述にエラーが生じることがあり、そのために、この段階でのテストが重要となる。

第２の問題は、ループ展開または定数伝搬／変数伝播のような典型的なコンパイラにより供給される高レベル最適化技術の欠如である。この問題は、単一の集積回路におけるトランジスタの数の増加およびオンチップシステム技術の出現にともない、Ｖｅｒｉｌｏｇコードの増加のために一層悪化することになる。これは、手動による最適化を行うために、ユーザーに長い時間を費やすことを強いる結果となる。

上述の問題から、抽象概念のレベルの向上が必要とされており、その対応する技術として高位合成（ＨＬＳ）がある。既知のＨＬＳツールとしては、I. Page and W. Luck によりCompiling Occam into FPGAs, 271-283, Abingdon EE and CS books, 1991に開示されているようなハンデル（Ｈａｎｄｅｌ）コンパイラおよびハンデルーＣコンパイラを有するものがある。ハンデル・コンパイラは、例えばInmos, The Occam 2 Programming Manual, Prentice-Hall International, 1988に開示されているような、オッカムとして知られている言語で書かれたソースコードを受けとる。オッカムは、Ｃに類似する言語であるが、並列処理および指定されたチャネルを介した同期２点間通信を表現するための余分な構成を有する。ハンデルＣコンパイラもほとんど同一であるが、原子言語が若干異なり、Ｃに慣れたプログラマにとって馴染み易いものとなっている。たとえば、プログラマは、各構成のタイミングの全体の制御を行う。各構成には、正確なサイクル数が割り当てられる（これは、時間付き意味規則（Timed Semantics）と呼ばれている。）。それゆえ、プログラマは、設計に際して全ての低レベル並列処理を考慮しなければならず、コンパイラが各構成をクロックサイクルにどのようにして割り当てるかを知っていなければならない。

しかし、全ての割り当てにはちょうど１サイクルかかるので、単一サイクルで起こるようにするには、両方の乗算を必要とする。これは、２つの乗算器が形成されなければならないことを意味し、面積を余分に要する。また、それらの乗算器は単一のサイクルで動作しなければならないので、クロックの速度が遅くなる。

上記問題を解決するコンパイラとして、抽象化のレベルをより一層高めたものがいくつか提案されている。それらのツールの多くは、最初にＨＬＳを実行し、次いでハードウェア・アプリケーション・ネットリスト・ファイルを発生するといった連続的

な設計手法を採用する。しかし、この場合は、利用可能な目的ハードウェアの面積またはアプリケーションのスループット仕様に適合しないことがある。そのような場合、設計フローの初めに正確なコンフィグレーション・オーバーヘッドおよびレイアウト・メトリクス（レイアウト指標）を提供できず、また初期の設計段階から設計の決定を撤回できないために、適切な解が見出されるまで処理が何回か繰り返される。

レイアウト・メトリクスを使用して上記処理の問題を解決した手法が提案されている。例えば、１９９９年３月１０日にスコットランドのグラスゴウで開催された、再構成可能システムについてのＩＥＥ専門家会議、ダイジェストＮｏ．９９／０６１に記載された、M. Vasilco、D. JibsonおよびS. Holloway著の「Towards a Consistent Design Methodology for Run-time Reconfigurable Systems」と題する論文、およびP. Lysaght著の「Towards an Expert System for a Priori Estimation of Reconfiguration Latency in Dynamically Reconfigurable Logic」（１８３～１９３ページの［３］）により開示されている。しかし、それらにおいて、メトリクス（指標）の正確な見積もりを行うには、設計アーキテクチャおよび構成スケジュール毎に設計モジュールの配置および詳細な配線が要求される。これは非常に簡単な設計であるが、非現実的であり、この理由から、それらのツールのほとんどが機能ユニット（ＦＵ）モデルのみを使用するものとなっている。このことは、より高い最適化を行う場合に、一層扱いづらいものとなり、以下のような困難な条件が要求されることとなる。

（１）面積／スループットの機能分担を最適にするために、各ＦＵで実行されるように結合した効率的なライブラリを必要とする。なお、アプリケーションのいくつかの部分が高速乗算器を要することがあるが、他の部分は低速乗算器で十分である。

（２）コードプログラム全体を共用する、すなわち、目的ＶＬＳＩ回路における基本ハードウェアセルの数の最大境界が与えられる効率的なＦＵを求める必要がある。これは、高いスループットを発生するために使用される各種ＦＵの最適な数である。

（３）ＣＡＤツールのほとんどはＦＵレベルで共用するハードウェアを考えて、マルチプレクサ用の大きなハードウェアを取っておく必要がある。これは、それらのマルチプレクサの価格が高いために、特にＤＲＬ／ＦＰＧＡ回路にとって重要である。

発明の概要

本発明の目的は、上述したような各問題を解決し、プログラマに馴染みの深い高級記述言語による電子回路モデルの記述が可能で、より正確なコスト見積もりを行うことができる、コンパイル方法および合成装置を提供することにある。

本発明のさらなる目的は、そのような設計を実行可能なプログラムを記録した記録媒体を提供することにある。

上記目的を達成するため、本発明のコンパイル方法は、所望の電子回路モデルが所定の高級記述言語で記述された記述ファイルを構文解析して所定のグラフ構造を有する制御データ・フロー・グラフを生成する第１のステップと、前記制御データ・フロー・グラフを、複数の連結されたノードの集合よりなる、特定の機能を果たすスレッドに分割し、該分割したスレッドを所定の面積制約および所定の待ち時間制約と合致するように最適化して、前記電子回路モデルに関する論理セルの数、機能、配置および配線の指定情報を得る第２のステップとを含むことを特徴とする。

上記の場合、前記第２のステップにおける最適化が、機能ユニット、レジスタ、マルチプレクサのいずれかに関する面積と待ち時間との最低境界を推定することにより行われてもよい。

また、前記第２のステップにおける最適化が、分割されたスレッドを所定の面積制約と合致するように最適化した後、さらにその最適化されたスレッドを所定の待ち時間制約と合致するように最適化するようにしてもよい。

さらに、前記第２のステップは、前記所定の面積制約および待ち時間制約に基づく最適化を最上位の分割スレッドから順に行うトップ・ダウン処理ステップと、前記トップ・ダウン・ステップにて最適化された下位の分割スレッドをいくつかのスレッドに分離して所定のコンテクストまたは所定の回路にまとめるダウン・トップ処理ステップとを含んでいてもよい。

上記の場合、前記トップ・ダウン処理ステップは、前記制御データ・フロー・グラフを、複数の連結されたノードの集合よりなる、特定の機能を果たすスレッドに分割する第１の分割ステップと、前記第１の分割ステップにて分割されたスレッドに対し

て所定の制御ステップおよび該ステップにおけるスレッドの移動レンジの割り付けを行うとともに、該制御ステップのおのおのに対して割り付けられたスレッドについて予め設定された複数の優先順位リストに従った優先順位を割り付ける第１のスケジューリング・ステップと、前記第１のスケジューリング・ステップによる割り付けが施されたスレッドについてトータル面積を見積り、該トータル面積が所定の面積制約に合致するか否かを判定する第１の面積制約判定ステップと、前記第１の面積制約判定ステップにて面積制約に合致しないと判定された場合に、前記第１の分割ステップにて分割されたスレッドの全てのスレッド対の組合わせについて面積に関する類似コストを算出する類似コスト算出ステップと、前記類似コスト算出ステップにて算出された類似コストを参照して、異なる制御ステップに属し、かつ、より高い類似コストを有するスレッド対を前記スレッド対のうちから選択し、該選択したスレッド対を新たなスレッドとして他のスレッドとを組合わせて新たなスレッド対を得る第１のアロケーション・ステップと、前記第１のアロケーション・ステップにて得られた新たなスレッド対についてトータル面積を見積って、該トータル面積が所定の面積制約に合致するか否かを判定する第２の面積制約判定ステップと、前記第２の面積制約判定ステップにて面積制約に合致しないと判定された場合に、前記複数の優先順位リストに従って、優先順位の低いリストから順に、リストに含まれているスレッドについて、同じ制御ステップに属し、かつ、より高い類似コストを有するスレッド対を選択し、該選択したスレッド対を新たなスレッドとして他のスレッドと組合わせて新たなスレッド対を得るとともに、該新たなスレッド対が割り付けられた制御ステップを同じ内容の２つの制御ステップに細分化するアロケーション－スケジューリング・ステップと、前記第１または第２の面積制約判定ステップにて面積制約に合致した場合に、前記第１のアロケーション・ステップまたはアロケーション－スケジューリング・ステップにて得られた新たなスレッド対について、前記面積制約と前記所定の待ち時間制約とのトレードオフを調べ、両制約に合致するように、ノードの配置および配線を行うスレッド処理ステップとを含み、前記ダウン・トップ処理ステップは、前記スレッド処理ステップにて配置配線されたスレッドについて、前記複数の優先順位リス

トに従って、優先順位の高いリストから順に、そのリストに含まれているスレッドのうちの類似性が低いスレッド対を選択して分離する第２のスケジューリング・ステップと、前記第２のスケジューリング・ステップにて分離されたスレッド対を、そのスレッドの間の結合性制約が最小となるようなコンテクストまたは回路にまとめる第２の分割ステップとを含むようにしてもよい。

本発明の合成装置は、所望の電子回路モデルが所定の高級記述言語で記述された記述ファイルを構文解析して所定のグラフ構造を有する制御データ・フロー・グラフを生成するフロント・エンド・コンパイラー手段と、前記制御データ・フロー・グラフを、複数の連結されたノードの集合よりなる、特定の機能を果たすスレッドに分割し、該分割したスレッドを所定の面積制約および所定の待ち時間制約と合致するように最適化して、前記電子回路モデルに関する論理セルの数、機能、配置および配線の指定情報を得るバック・エンド・コンパイラー手段とを有することを特徴とする。

上記の場合、前記バック・エンド・コンパイラー手段は、機能ユニット、レジスタ、マルチプレクサのいずれかに関する面積と待ち時間との最低境界を推定することにより前記最適化を行うように構成されてもよい。

また、前記バック・エンド・コンパイラー手段は、前記制御データ・フロー・グラフを、複数の連結されたノードの集合よりなる、特定の機能を果たすスレッドに分割する第１の分割手段と、前記第１の分割手段にて分割されたスレッドに対して所定の制御ステップおよび該ステップにおけるスレッドの移動レンジの割り付けを行うとともに、該制御ステップのおのおのに対して割り付けられたスレッドについて予め設定された複数の優先順位リストに従った優先順位を割り付ける第１のスケジューリング手段と、前記第１のスケジューリング手段による割り付けが施されたスレッドについてトータル面積を見積り、該トータル面積が所定の面積制約に合致するか否かを判定する第１の面積制約判定手段と、前記第１の面積制約判定手段にて面積制約に合致しないと判定された場合に、前記第１の分割手段にて分割されたスレッドの全てのスレッド対の組合わせについて面積に関する類似コストを算出する類似コスト算出手段と、前記類似コスト算出手段にて算出された類似コストを参照して、異なる制御ス

テップに属し、かつ、より高い類似コストを有するスレッド対を前記スレッド対のうちから選択し、該選択したスレッド対を新たなスレッドとして他のスレッドとを組合わせて新たなスレッド対を得る第１のアロケーション手段と、前記第１のアロケーション手段にて得られた新たなスレッド対についてトータル面積を見積って、該トータル面積が所定の面積制約に合致するか否かを判定する第２の面積制約判定手段と、前記第２の面積制約判定手段にて面積制約に合致しないと判定された場合に、前記複数の優先順位リストに従って、優先順位の低いリストから順に、リストに含まれているスレッドについて、同じ制御ステップに属し、かつ、より高い類似コストを有するスレッド対を選択し、該選択したスレッド対を新たなスレッドとして他のスレッドと組合わせて新たなスレッド対を得るとともに、該新たなスレッド対が割り付けられた制御ステップを同じ内容の２つの制御ステップに細分化するアロケーション－スケジューリング手段と、前記第１または第２の面積制約判定手段にて面積制約に合致した場合に、前記第１のアロケーション手段またはアロケーション－スケジューリング手段にて得られた新たなスレッド対について、前記面積制約と前記所定の待ち時間制約とのトレードオフを調べ、両制約に合致するように、ノードの配置および配線を行うスレッド処理手段と、前記スレッド処理手段にて配置配線されたスレッドについて、前記複数の優先順位リストに従って、優先順位の高いリストから順に、そのリストに含まれているスレッドのうちの類似性が低いスレッド対を選択して分離する第２のスケジューリング手段と、前記第２のスケジューリング手段にて分離されたスレッド対を、そのスレッドの間の結合性制約が最小となるようなコンテクストまたは回路にまとめる第２の分割手段とを有する構成としてもよい。

本発明の記録媒体は、所望の電子回路モデルが所定の高級記述言語で記述された記述ファイルを構文解析して所定のグラフ構造を有する制御データ・フロー・グラフを生成する処理と、前記制御データ・フロー・グラフを、複数の連結されたノードの集合よりなる、特定の機能を果たすスレッドに分割し、該分割したスレッドを所定の面積制約および所定の待ち時間制約と合致するように最適化して、前記電子回路モデルに関する論理セルの数、機能、配置および配線の指定情報を得る処理とをコンピュー

タに実行させるプログラムを記録したことを特徴とする。

上記のとおりの本発明は、ハードウェアシステムに対する新しいＣＡＤ設計技術を提供する。その主要な点は、高レベル合成ツールと低レベル合成ツールの間のギャップを埋めることにある。入力言語を高レベルでプログラマになじんだものにでき、かつハードウェアにおいて理解できる表現を有する重要な構成のほとんどをサポートできる。

本発明においては、まず、比較的高いレベルで最適化が行われ、制御データフローグラフ（ＣＤＦＧ）が出力される。そして、ＣＤＦＧは、スレッドと呼ばれる、結合されたノードの独立したクラスタに分割される。このやり方により、スケジューリング、アロケーションおよび分割が、単一オペレーションレベルではなくて、スレッドレベルで行われることになる。これは、特にＦＵ遅延がユーザークロックサイクルより比較的短い場合に、高いスループットをシステムに与えることに加えて、ＨＬＳの複雑さを減ずる。また、そのようなスレッドのおのおのに対して、コンパイラの最上ステージの間に、ＦＵ、レジスタおよびマルチプレクサについて面積および待ち時間の最低境界推定を同時に行える。さらに、その最後のステージの間に、配置配線コストとを考慮することにより、より正確なコスト見積もりを行える。

また、本発明によれば、高い性能／面積のトレードオフを達成するために、ライブラリ結合が効率的に行われる。

さらに本発明では、スレッドは、少なくとも１つの分岐の深さが所定のしきい値を超えている場合に、結合されているノードにより形成されて、ＩＯポートを共用している２つの連続するメモリアクセスまたはＩＯアクセスの間に見出されるブロックとして、またはユーザーにより導入される明示機械として、あるいは制御グラフのフォークジョインノードとして定められる。そして、スレッドは、最初の分割中に抽出される。これにより、単純なノードではなくて、スレッドの群に高レベル合成が適用されることになり、コンパイラ実行時間を短縮することが可能になる。更に、そのような独立したスレッドは、ライブラリの結合、配置配線中に効率良く利用される。

また、本発明では、各スレッドに対して面積／待ち時間のトレードオフを考慮する

とともに、ライブラリの結合と近さ（ｃｌｏｓｅｎｅｓｓ）とのコストが加えられる。配線長を最も短くするために、スレッド処理中に近さの距離が調べられる。これは、配線遅延がハードウェアセルの遅延よりも一層重要であることから、超サブミクロン技術に特に有効である。

また、本発明におけるフロント・エンド・コンパイラーでは、待ち時間の計算にハードウェアセル、レジスタ、マルチプレクサの遅延を考慮することが可能である。設計フロー・ツールの最終段階において、クリティカルパスを考慮することにより、遅延制約に関する確度が高くなる。

さらに、設計に用いるコストとしては、一致性（ｃｏｎｃｕｒｒｅｎｃｙ）条件、類似性条件、結合性（ｃｏｎｎｅｃｔｉｖｉｔｙ）条件、および分岐条件を用いることが可能である。本発明では、これらのコストが反復的に用いられることで、処理の効率化が図られる。ここで、アロケーション中に、一致性コスト／パイプラインコストにしたがって、システムのスループットに円滑に影響を及ぼさせるようにするのが類似性である。その後で、チップの間の相互接続を最小にするために、またはＤＲＬの場合にレジスタの数を減少するために、結合性メトリク（ｃｏｎｎｅｃｔｉｖｉｔｙ　ｍｅｔｒｉｃ）が用いられる。

さらに本発明では、制御ステップを各スレッドに割り付けるにあたって、スレッドは優先度リストにしたがって配置される。そして、スレッドの移動レンジ、スレッド生存期間、分岐条件、並列スレッド、パイプライン化されたスレッドとが条件として考慮されて最適化が行われる。

ハードウェアセルの数が十分でないとした場合は、類似性コストが計算される。対応するデータ構造をマトリクスで構成することができる。スレッドが２つ以上のＦＵを共用する場合、アロケーションにより、マルチプレクサの数を最小にすることができ、その結果、スレッド待ち時間が短くなる。

アロケーションには、マルチプレクサまたは種々のコンテキストを使用することできる。また、アロケーションは、同じセグメントステップに属していないスレッドに対してのみ実行される。

アロケーション－スケジューリングは、設計装置のスループットを徐々に増加するために用いられる。このアロケーション－スケジューリングでは、最低優先度リストに属し、かつ最高類似性メトリクスを有するスレッドが実行される。その後で、対応する制御ステップが２つに分割される。そして、面積が見積もられ、そのリストの全ての要素が処理されるまでプロセスが繰り返される。このアロケーション－スケジューリングの後に、スレッド処理が行われる。面積を更に減少して、ループに属し、かつ待ち時間制約に合致しないスレッドに対するハードウェアパイプラインを最終的に発生する。これは、スレッド調整と、スレッド最適化との２つのステップを含む。スレッド調整では、面積／遅延評価のためにハードウェアセルモデルと、マルチプレクサモデルと、レジスタモデルとを使用することができ、これにより、全てのスレッドの待ち時間制約が確保される。

タイミング解析には、各スレッドの待ち時間を正確に評価するためにＥｌｍｏｒｅ遅延モデルを使用することが可能である。待ち時間制約に合致しないものは、求められている数のレジスタをそれのノードの間に挿入することによって分割される。この段階で、待ち時間制約に合致する間に、面積を更に減少するために、ライブラリ結合が実行される。ライブラリ結合では、同じ種類のＦＵの種々のバージョンを使用することができる。これは、他の高レベル合成システムに対しては明らかなタスクではないため、その場合には同じ種類のＦＵが一般に用いられる。

図面の簡単な説明

図１は、本発明のコンパイル方法の一実施形態である高レベル設計フローの一例を示すフローチャート図である。

図２ａ及び図２ｂは、高水準入力記述ファイルの一例を示す図である。

図３は、図１に示す設計フローに用いられるＬＳＩ回路の例を示すブロック図である。

図４は、図１に示す高レベル設計フローを適用する汎用コンピュータ装置の一構成例を示すブロック図である。

図５は、図１に示すバック・エンド・コンパイラーにおける処理の流れを示すフロ

ーチャート図である。

図６は、映像および音声のデータの入出力が可能な記憶装置を構成する目的ハードウェアの構成例とそれに関する複数のアプリケーションの記述例を示す図である。

図７は、図６に示す目的ハードウェアおよびアプリケーションを図５に示す処理に適用した時のタイルを横切るメモリ分配の結果を示す図である。

図８は、図６の例に対するスレッド抽出結果を示す図である。

図９は、図６の例に対する類似性コスト測定結果を示す図である。

図１０は、図６の例に対する第１のスケジューリング結果を示す図である。

図１１ａ及び図１１ｂは、ＤＲＬの場合におけるＦＵ共用のアロケーション形態を示す模式図である。

図１２ａ、図１２ｂ及び図１２ｃは、図６の例に対するアロケーション－スケジューリング結果を段階的に示す図である。

図１３ａは、移動レンジ制約を模式的に示す図である。

図１３ｂは、スレッド共有制約を模式的に示す図である。

図１３ｃは、パイプライン制約を模式的に示す図である。

図１４は、スレッド調整手順の一例を示すフローチャート図である。

図１５は、スレッド最適化手順の一例を示すフローチャート図である。

図１６は、図１２ａ、図１２ｂ及び図１２ｃに示すスレッドのノードクラスタリング実行結果である、ラベル割り付けおよびクリティカルパスの一例を示す図である。

図１７は、図１２ａ、図１２ｂ及び図１２ｃに示すスレッドのノードクラスタリング実行結果である、近さマトリクスの計算例を示す図である。

図１８は、図１２ａ、図１２ｂ及び図１２ｃに示すスレッドのノードクラスタリング実行結果である、クラスタツリーの一例を示す図である。

図１９は、図１２ａ、図１２ｂ及び図１２ｃに示すスレッドのノードクラスタリング実行結果におけるレジスタ・リタイミングの組合わせ例を示す図である。

実施形態

次に、本発明の実施形態について図面を参照して説明する。

図１は、本発明のコンパイル方法を適用した高レベル設計フロ--の一例を示す。この高レベル設計フローは、コンピュータを使用して回路設計を行うシステム（ＣＡＤ）において行われる処理であり、破線で囲まれた部分が論理合成を行うシステム１０７における処理を示す。この合成システム１０７における処理は、ＡＳＩＣ、ＦＰＧＡなどの回路またはＤＲＬなどの論理回路の作成に適用することができる。

まず、ユ--ザー１０１が，高水準入力記述ファイル１０２を入力して、対話処理（人間が端末を介して計算機に指示を与えながら問題解決を行うデ--タ処理）を行う。高水準入力記述ファイル１０２はテキスト形式であって、その記述にはＪａｖａ、Ｃ、またはＣ＋＋などの既存の高級言語を用いることができる。また、この高水準入力記述ファイル１０２は、そのような言語には含まれないいくつかのハードウェア拡張を支援することができる。

図２ａ及び図２ｂに、そのようなハードウェア拡張例を示す。図２ａ及び図２ｂに示すハードウェア拡張２０１、２０４は、上述の高水準入力記述ファイル１０２中に記述される、ハードウェア拡張を支援するための記述である。ここでは、一例としてＩＯポート仕様２０２、メモリ仕様２０３、明示状態機械挿入２０５、ビットレベル操作２０６のハードウェア拡張例が示されている。ＩＯポート仕様２０２では、変数ｃとｘが回路の入力ポートと出力ポートにそれぞれ割り当てられる。この場合、ピン番号割り当ても支援することができる。メモリ仕様２０３では、変数ｄ１とｄ２が９ビットデータ幅の二次元メモリに割り当てられる。明示状態機械挿入２０５では、条件付き変数ｃが１に等しい場合には、１アイドル・サイクルが変数ｘを決定する前に挿入され、等しくない場合には、後に挿入さる。この拡張では、相互タスク同期機構を特徴とする、Ｊａｖａのような言語は要求されない。ビット・レベル操作２０６では、ビット・レベルの指定が可能となっている。

フロント・エンド・コンパイラー１０３は、パラメータを持つ機能と機能呼出しとを伴う表現を含むほとんど全ての言語構文を処理することができる。よって、このフロント・エンド・コンパイラー１０３の処理により、そのような言語に既に慣れているソフトウエア開発者の作業を容易にし、しかも、ソフトウエア開発者に対して、ハ

ードウェアに関する十分な知識が要求されることもない。また、フロント・エンド・コンパイラー１０３は、入力記述ファイル１０２を構文解析してその中間フォーマットである制御データ・フロー・グラフ（ＣＤＦＧ）１０４を出力する。

バック・エンド・コンパイラー１０５は、フロント・エンド・コンパイラー１０３によって構文解析されたデータ構造に最適化などの処理（詳しくは、後述する）を施して、ハードウェア・アプリケーション・ネットリスト・ファイル１０６を生成する。このバック・エンド・コンパイラー１０５は、インタフェースでサーバーと接続された、モジュールライブラリ１１０を含むマネジャーの役割を果たす。ハードウェア・アプリケーション・ネットリスト・ファイル１０６は、使用されている論理セルの数、それらの機能、配置および配線要求などの指定情報を含む。さらに、このファイル１０６は、マルチ・コンテクストＤＲＬおよびマルチ・チップ・ハードウェアのそれぞれの場合において、それらの割り付けられたコンテクストまたはチップの情報を含む。モジュールライブラリ１１０は、対応する面積と遅延を有する、各種のパラメータの設定が可能な機能ユニット（ＦＵ）の集合を供給する。

上記バック・エンド・コンパイラー１０５における最適化処理では、ハードウェア制約１０９（面積制約）及び時間制約１０８（待ち時間制約）が考慮される。例えば、図１に示した設計フローにおいて、生成した全ハードウェア量（例えばセル、処理装置、またはトランジスタの数）が利用可能なハードウェアの量以下の場合に、ハードウェア制約１０９が確認され、また、生成したサイクル数が要求される数以下の場合に時間制約１０９が確認される。

図３に、図１に示した設計フローを適用できる各種ＬＳＩ回路を示す。

図３（ａ）は、ＡＳＩＣまたはＦＰＧＡのデバイスの構成を示す模式図である。図３（ａ）において、ＡＳＩＣまたはＦＰＧＡのデバイス３０１は、制御バス３０２と、データ・バス３０３と、任意の埋め込みメモリ３０４との組み合わせにより構成される。

図３（ｂ）は、ＤＲＬの構成を示す模式図である。図３（ｂ）において、動的ＤＲＬ３０５は、基準構成を含む複数のコンテクスト３０６ａ～３０６ｄとアクティブ・

プラン３０７とで構成されている。この構成では、１度に１つのコンテクストがアクティブ状態となる。

図３（ｃ）は、マルチ・チップ回路の構成を示すブロック図である。図３（ｃ）において、マルチ・チップ回路３０８は、相互接続ネットワーク３１０を介して相互に接続された複数の回路３０９から構成されている。

以上説明した本形態の高レベル設計フローは、例えば図４に示すような汎用コンピュータ装置を用いて実現することができる。図４において、汎用コンピュータ装置４００は、図形情報とテキスト情報を表示するための図形スクリーン４０１を備える図形表示モニタ４０２と、情報のテキスト入力のためのキーボード４０３と、コンピュータ・プロセッサ４０４と、コンパイル・プログラムを記録した記録媒体４０５からなる。コンピュータ・プロセッサ４０４は、キーボード４０３および表示モニタ４０１に接続されている。本例では、コンピュータ・プロセッサ４０４に、記録媒体４０５から上述した高レベル設計フローを実現するためのプログラム・コードが与えら、これによる後述する種々のコンパイル処理が実行される。この汎用コンピュータ装置４００には、メインフレーム・コンピュータ、ミニ・コンピュータ、パーソナル・コンピュータなど、良く知られている種々のタイプのコンピュータを用いることが可能である。記録媒体４０５は、磁気ディスク、半導体メモリ、その他の記録媒体であってよい。

次に、バック・エンド・コンパイラー１０５における処理について詳細に説明する。

図５は、バック・エンド・コンパイラー１０５における処理の流れを示すフローチャート図である。このバック・エンド・コンパイラー１０５における処理は、２つのフェーズ、すなわち、トップ・ダウン・フェーズ５０２と、ダウン・トップ・フェーズ５０３とに分けることができる。

トップ・ダウン・フェーズ５０２では、まず、いくつかの性質を特徴とする結合ノードを独立スレッドに分類するために、第１の分割であるスレッド抽出５０４の処理が行われる。ここで、結合ノードは、グラフにおける枝によって接続されたノードで

あり、スレッドは、それら結合ノードの独立クラスタ（複数の連結ノードの集合よりなる、特定の機能を果たすモジュールの組み合せ）である。スレッドが抽出されると、続いて、その抽出されたスレッドについて、面積制約が合致するまで、スケジューリング５０５、スレッド類似性５０６、アロケーション５０７、アロケーション－スケジューリング５０８の処理が段階的に行われ、これにより最適化が行われる。これら各段階の処理については、後述の実施例で詳細に説明する。

上記の最適化が行われた後、得られたスレッドのそれぞれは、独立したやり方で、低位合成モジュールであるスレッド処理５０９をモジュール・ライブラリー５１２から呼び出すことによって処理される。このスレッド処理５０９では、図１に示した時間制約１０８が確認される。この処理の目的は、待ち時間制約が各スレッド毎に合致することを保証することにある。このタスクは、レイアウト・レベルでの面積／コストの指標を使用するので、正確である。

他方、ダウン・トップ・フェーズ５０３では、スレッド処理された各スレッドについて、システムのスループットを一層増大するための処理が行われる。すなわち、このダウン・トップ・フェーズ５０３では、第３のスケジューリング５１０にて、上述のトップ・ダウン・フェーズ５０２の第２のスケジューリング５０８にて組合わされたいくつかのスレッドを分離する処理が行われ、最後に、第２の分割５１１にて、その分離されたスレッドをＤＲＬ用の種々のコンテクスト、またはマルチチップ・ハードウェア用の種々の回路にまとめる処理が行われる。

＜＜実施例＞＞

次に、上述のバック・エンド・コンパイラーの処理について、さらに詳細に説明する。ここでは、より動作を分かり易くするため、マルチスレッド・アプリケーションを考慮した実際のやり方について説明する。

図６に、映像および音声のデータの入出力が可能な記憶装置を構成する目的ハードウェアの構成例とそれに関する複数のアプリケーションの記述例を示す。この図６の例は、映像処理アプリケーションと音声処理アプリケーションとの組合わせを考慮した例である。第１のアプリケーションである動作推定６０１と、第２のアプリケ-

ションである２－Ｄ有限インパルス（ＦＩＲ）フィルタ６０２とは、同時に実行することが可能である。第３のアプリケーションである自己相関フィルタ６０３は、音声信号上に自己相関を適用する。全ての入力データは、８ビット幅となることが前提とされる。バック・エンド・コンパイラーの処理の目的は、最小のハードウェアで可能な限り高いスループットを達成することである。この例では、目的ハードウェア６０４は、１２個の入力ポートと３つの出力ポートを有しおり、各アプリケーションに関して、４つの入力データを同時に処理することができる。

以下、前述の図５におけるバック・エンド・コンパイラーの処理を上記図６の構成例に適用した場合の各処理について具体的に説明する。

（第１の分割）

第１の分割であるスレッド抽出５０４では、ＣＤＦＧから結合ノード群であるスレッドが抽出される。このスレッドは、結合ノード群より形成されたブロックとして定義され、少なくとも１つの分岐の深さが所定のしきい値を超えた場合に、同じＩＯポートを共有している２つの連続するメモリまたはＩＯアクセスと、ユーザーにより導入される明示状態機械と、明示スレッド・プロセスと、制御グラフの分岐結合ノードとに基づいて抽出される。この処理により抽出されるスレッドは、その大きさ（連結ノードの数）により以下のような条件とすることが望ましい。

（ａ）ハードウェア合成（技術マッピング、配置および配線）の低位部分の時間的複雑さを減ずるように、スレッドは十分に小さくする。その理由は、ＩＯアクセスがプログラム、特にマルチメディア・アプリケーションにおいて頻繁に起きるためである。

（ｂ）単一の操作よりむしろ、スレッドの最適化を行うことによりＨＬＳの高位部分の時間的複雑さを減ずるように、スレッドは十分に大きくする。これは、一連の複数の操作を同時に行うことによって、プロセッサ／ＤＳＰよりもＦＰＧＡに対する利点を高める。その理由は、それら複数の操作に加えて、同時に起こるＩＯアクセスを同じスレッドにまとめるようにＡＳＩＣ、ＦＰＧＡおよびＤＲＬのいくつかのＩＯポートを含むためである。加えて、十分な数のレジスタが目的ＶＬＳＩ回路によって提

供されるために、中間変数が外部メモリよりはむしろ内部レジスタに保存されるからである。更に、前述した本形態の設計ツールはそれらの中間変数の寿命時間を減少するのに役立つからである。

メモリアクセスを含むループの場合には、繰り返し中にメモリ並列が存在するかどうかを決定するために、データおよびループ拡張従属性が与えられなければならない。この目的は、メモリ並列を調べるために静的アクセスの特定をヒング（hings on）することにある。これは、一緒に頻繁にアクセスされるデータをメモリ、すなわち、タイル（ｔｉｌｅｓ）を横切って分配することにより行われる。この場合、アレイは低順位のインターリーブ順位（主記憶を同時にアクセス可能な複数個のモジュールに分割して構成する場合にモジュールを横切ってアドレスをつけるインターリービングにおける各モジュールの順位に相当する。）で一様に配列される。つまり、データ構造の連続要素が、連続するタイルを横切ってラウンドロビン法でインターリーブされる。このレイアウトは、空間的な閉アレイ・アクセスが仮に閉じられるので望ましい。次に、コードを変換して並列アクセスを可能にするために、ループが繰り返される。

図７に、上述の図６の例を図５の処理に適用した時の、タイルを横切るメモリ分配の結果を示す。入力フレームメモリ７０１、７０２および入力音声メモリ７０３、７０４に対して４つのタイルが得られる。入力フレームメモリ７０１、７０２、入力音声メモリ７０３、７０４は、図６に示した入力フレームメモリ６１１、６１２、入力音声メモリ６１３、６１４にそれぞれ対応する。

図８に、結果として得られた３つのスレッドを示す。スレッド８０１、８０２、８０３はそれぞれ図６に示した動作推定６０１、有限インパルス（ＦＩＲ）フィルタ６０２、自己相関フィルタ６０３の各アプリケーションに対応する。これらスレッド８０１、８０２、８０３は、それぞれ図６に示した目的ハードウェア６０４の「Ｄｉｆｆ」、「Ｏｕｔ」、「Ｒｅｓｕｌｔ」の出力とされ、それぞれ出力フレームメモリ６１５、６１６、６１７に格納される。

（第１のスケジューリング）

次に、上述の第１の分割で抽出されたスレッドについて、第１のスケジューリングが行われる（図５の５０５の処理）。この目的は、各スレッドにそのＡＳＡＰ（As Soon As Possible）制御ステップ値およびその移動レンジを割り付けることにある。設計フローの更なる考慮のために、優先順位で配置されているスレッドのリストがそれらのステップのおのおのに割り付けられる。たとえば、以下のような優先リストを考慮した割り付けができる。

（ａ）Ｐリスト１：所定のしきい値（実時間ＩＯアクセスのような）を超えない移動レンジを持つスレッドのリストである。

（ｂ）Ｐリスト２：活性が所定のしきい値より小さくないスレッドからなる。

（ｃ）Ｐリスト３：ループに属するスレッドを含み、その先行値は既にスケジュールされている。

（ｄ）Ｐリスト４：分岐条件に対応するスレッドのリストである。

（ｅ）Ｐリスト５：ユーザーにより明示的に定められたパイプライン／並行スレッドから構成されている（ｊａｖａにおけるマルチスレッディング）。

（ｆ）Ｐリスト６：すぐ後に続く要素を有するスレッドのリストである。

（ｇ）Ｐリスト７：残りのスレッドを構成する。

まず、現在の制御ステップ中の全てのノードに対してスケジュール処理を施すためにＰリスト１が考慮される。さもなければ、それらの移動度を遅らせることはスケジューリングを引き伸ばすことになる。したがって、移動度は良い優先関数と考えられる。

次に、さらなるスレッドをロードする。より迅速なハードウェア可用度にすることによって、再構成オーバーヘッドを減少するためにＰリスト２がＤＲＬ回路に独占的に考慮される。また、高い優先度では、先行値が既にスケジュールされているループのストリームがスケジュールされる。これは、中間レジスタの数を減少するため、および同じループ内のコンテクスト・スイッチを減少するために行われる。次に、分岐条件に対応する条件が解決される。これは、それらの分岐のノードをスケジューリングするより多くの選択肢を与える。

図９に、図５の例に適用した場合のアルゴリズムの結果を示す。音声処理のサンプリング・レート(-)がビデオ信号のそれのｐ倍であると仮定する。その場合、実際のステップが異なると、スレッド１および２の移動度レンジが１クロック・サイクルであるので、スレッド１および２はＰリスト１において同じ順位で順位付けされる。その他、スレッド３がＰリスト７に加えられる（図９）。

（類似性コスト）

第１のスケジューリング（図５の５０５）が行われた後にトータル面積が見積もられ、見積もったトータル面積が面積制約に合致するかが判定され、合致した場合は、十分なハードウェアセルであるとして後述するスレッド処理が行われ、合致しなかった場合は、スレッド対の全ての組合わせに対して類似メトリクス（図５の５０６）が計算される。そして、マトリクス、類似マトリクスが推測される。コストは、次の処理のいずれかにより最良の組合わせを達成した後の得られる面積の削減に対応する。

（ａ）２つの異なるスレッドの対を組合わせる。このコストは、第１のアロケーション（図５の５０７）、またはアロケーション－スケジューリング（図５の５０８）の間のいずれかで用いられる。

（ｂ）同じスレッドを分割する。対応するコストが、その後に続くコンパイラのステップの間に用いられるたびに更新される。この場合は、アロケーション－スケジューリング（図５の５０８）の間に独占的に考慮される。

ここで、類似コストについて、さらに具体的に説明する。面積としてエリア１、エリア２をそれぞれ有する２つのスレッド１、２について考える。この場合の類似コストは、

類似コスト＝エリア１＋エリア２－エリア１２

である。ここで、エリア１２は、スレッド１とスレッド２を結合した後の、結果としての面積である。他方、同じスレッド１を分割する場合の類似コストは、

類似コスト＝エリア１＋新エリア１

である。ここで、新エリア１は、スレッドの分割後に得られた新しい面積である。図５の例を参照し、各ビットの動作が１つのハードウェアセルにより行われると仮定す

ると、図８に示したスレッド８０１、スレッド８０２、スレッド８０４の面積は、それぞれ７１、４４９、７１となる。

図１０に、図６に示した例についての第１のスケジューリング結果を示す。この例は、スレッド１とスレッド２を結合したものであり、その類似コストは目的ＶＬＳＩ回路に大きく依存する。その理由は、マルチプレクサによって要求されるハードウェアの量が、他の演算／論理ＦＵの量と比較した場合に、ＡＳＩＣに対する場合よりも、ＤＲＬ／ＦＰＧＡ回路に対する方がより重要であるからである。ＡＳＩＣの場合には、ハードウェアを一層削減できる。

（第１のアロケーション）

類似コストを用いて、第１のアロケーション（図５の５０７）が行われる。その役割は、制御ステップ間でＦＵの最大数を共有することにある。これは、マルチプレクサの数およびコードサイズを削減するという利点があり、埋め込みアプリケーションにとって重要である。その原理は、異なるステップに属し（同時性に影響しないようにするために）、かつ、より高い類似コストを有するスレッドの対を類似マトリクスから選択し、それらを新しいスレッドと組合わせることである。そして、その対は、面積制約に合致するまで、または異なる制御ステップに属する全てのスレッド対が処理されるまで繰り返されるプロセス、およびマトリクスから除去される。類似ノードの降下順によるそれら対の考慮には、全体の面積をより良く調べるという利点がある。そして、ＤＲＬ回路の場合には、図１１ａ及び図１１ｂに示すような２つのアロケーションの形態をとることが可能である。結果として得られるスレッド対は、マルチプレクサ１１０１を用いてそれらのスレッドの類似ブロックを共用でき、あるいはコンテクストスイッチを用いた２種類のコンテクスト１１０２としてマップできる。後者の場合は、マルチプレクサが使用されないので、面積および待ち時間の表現において、より高い性能を提供できる。しかし、制限付きコンテクスト数に関して、次のようないくつかの制約が適用される。

（ａ）スレッド間の結合性：

高結合スレッドは、同じコンテクストでマップされるために一層良い。ところが一

方、低く相互に結合されたものは異なるコンテクストにマップされる。これは、レジスタの使用を最少にするためである。

（ｂ）マルチプレクサによって生じる遅延時間を削減するための同じパス中のばらばらの類似ブロックの数。

（ｃ）コンテクスト・スイッチを避けてそのようなことが頻繁に生じるためのスレッド間の制御ステップの数。

（アロケーション－スケジューリング）

上記第１のアロケーション（図５の５０７）のアルゴリズムが面積制約に十分に合致しなかった場合には、アロケーション－スケジューリング５０８が、同じ制御ステップに割り当てられているスレッド上で実行される。その後、新しいステップが徐々に作られる。

まず、初めに、最も低い優先度（Ｐリスト７）を持つリストに属し、かつ最も高い類似メトリクスを持つスレッドが実行される。そして、対応する制御ステップが２つの制御ステップに細分化される。面積が見積もられ、そのリストの全ての要素が処理されるまでプロセスが繰り返される。次の優先度リスト（Ｐリスト６）に属するスレッドが同様に処理される。このような処理が、最も高い優先度リスト（Ｐリスト１）が処理されるまで繰り返される。

ここで、重要なことは、１つのスレッドが選択されると、ただ１つの追加状態が対応する制御ステップに挿入されることである。すなわち、リストを考慮する場合、最高優先度のリストに属するスレッドの間では、スレッドの共有は許されない。

図１２ａ、図１２ｂ及び図１２ｃに、図５の例に適用した場合の、上記コンパイラのアロケーション－スケジューリングの段階の結果を示す。

図１２ａは、第１の反復結果（面積＝４８１）である。スレッド８０３’は、Ｐリスト７に属するので、最初に選択される。この選択されたスレッド８０３’が最良の類似コストを表わすスレッド８０２’と組合わされて新しいスレッド１２０１を構成する。対応するスケジューリング１２０２では、新しい状態１２０３が挿入される。

図１２ｂは、上記第１の反復に続く第２の反復の結果（面積＝３６８）である。こ

の反復中、Ｐリスト１が考慮される。類似コスト・マトリクスによれば、最良の類似コストは（スレッド２，スレッド３’）である。対応するスレッド１２０４は、より少ないハードウェアで済む。この場合、スケジューリング１２０５のような制御ステップのスケジュールとなる。

図１２ｃは、上記第２の反復に続く第３の反復の結果（面積＝３２１）である。この反復では、スレッド１がＰリスト１から次の候補として実行され、スレッド１２０４と組合わされて、スレッド１２０６を発生する。この場合、スケジューリング１２０７のような制御ステップのスケジュールとなる。

（スレッド処理）

この作業は、２つの主ステップに細分化される。まず、後述のスレッド調整中に、待ち時間／面積のトレードオフが各スレッド毎に調べられる（第１のステップ）。このとき、ＦＵ、レジスタ、マルチプレクサの各遅延を考慮する。スレッド最適化と呼ばれる第２のステップは、各スレッドを、配線分布が最適となる方法で、目的ハードウェアの相関的な位置に物理的にマップする。この段階の低位合成では、配線遅延情報を取り入れることができる。

（１）スレッド調整（第１のステップ）：

この段階では、各スレッドは、それらスレッドを待ち時間制約と合致させるために、独立した方法で処理される。具体的には、図１３ａ、図１３ｂ及び図１３ｃに示すような、次の３つの異なるケースで取り扱われる。

（ａ）移動レンジ制約：

例えば、図１３ａに示す移動レンジ制約１３０１では、スレッド１３０３の待ち時間（ｔk2－ｔk1）は、その移動レンジ１３０２のそれより小さくなければならない。

（ｂ）スレッド共有制約：

図１３ｂに示すスレッド共有制約１３０４では、いくつかのＦＵを共有するスレッドは、時間的に重なり合ってはならない。スレッド１３０５とスレッド１３０６が、いくつかのＦＵを共有し、かつ、（ｔk2－ｔk1）と（ｔk4－ｔk3）がそれらの待ち時間をそれぞれ表しているものと仮定する。そして、スレッド調整は、（ｔk2－ｔk1）

が（ｔk4－ｔk3＋ｔMob）より小さくなるように確実にしなければならない。ここで、ｔMobはスレッド１３０６の移動レンジである。

（ｃ）パイプライン制約：

図１３ｃに示すパイプライン制約１３０８では、ループに属するスレッド１３０９の待ち時間は、所定の値を超えてはならない。この制約は、各スレッドに対するパイプライン段階の数を最少にするために用いられる。

図１４は、スレッド調整のフローチャート図である。まず、スレッドが実行される（１４０２の処理）。その待ち時間が上記制約の１つに合致しない場合（１４０３の処理）、アルゴリズムがそのような制約に合致する最小スレッド面積に対応する最良の解を見出す。これは、ライブラリ結合により行われる（１４０４の処理）。これと同じ操作が、全てのスレッドに対して行われる（１４０５の処理）。最後に、結果としての全体の面積が見積もられ（１４０６の処理）、それが利用可能なハードウェア面積よりも大きい場合には、図５に示したアロケーション５０７およびアロケーション－スケジューリング５０８が実行される。この処理は、面積／待ち時間制約に合致するまで、組合わせ操作によって影響を受ける全てのスレッドについて繰り返される。各スレッドのサイズが比較的小さいために、ライブラリ結合が速く達成される。

（２）スレッド最適化（第２のステップ）

ここでの目的は、効率的なやり方で各スレッドのノードの間の配置および配線を行うこと、および各スレッドの面積／待ち時間メトリクスの正確さを向上することである。

図１５に、スレッド最適化のフローチャート図を示す。まず、スレッドが実行される（１５０２の処理）。その独特の優先度として結合性制約を用いて、アルゴリズムが各スレッドのクリティカルパス（ＬＳＩの回路ブロックの入力端子から出力端子までの全信号経路の中で、信号伝搬遅延が最大となる経路のこと。）を調べ、ノード・クラスタリング（パターン空間に類似性の尺度を導入してパターンの標本ベクトルを似たものどうし集めて類に分けること。）フェーズの間にそのノードをクラスタ（各層は複数のユニットで構成され、入力層以外の層は通常複数のクラスターと呼ばれる

ユニットの集団に分割さている。）にまとめる（１５０３の処理）。待ち時間が１クロック・サイクルより長いスレッドの場合には、スレッドに挿入すべきレジスタの数がその後で演算される。その後、ライブラリ結合を通じて、スレッド面積を一層小さくするために（１５０５の処理）、レジスタ・リタイミングが実行される（１５０４の処理）。この作業は、最小面積に対応し、待ち時間制約に合致する解が見出されるまで、繰り返される（１５０６の処理）。

以下に、ノード・クラスタリングおよびレジスタ・リタイミングの処理を簡単に説明する。

（ａ）ノード・クラスタリング：

この段階の主な目的は、最適なやり方でノードをグループ化することにある。データパス回路に以下のような階層を定める。

（ａ－１）基本ブロック：ユニットセルのクラスタ。セルはローカル相互接続ネットワークを介して相互に接続されるので、それは低い伝播遅延（論理回路の入力パルスに対する出力パルスの平均の遅れ）の属性を持つ。

（ａ－２）マクロブロック：近似基本ブロック（closest elementary block）の集合。伝播遅延制約に依存して、それらの基本ブロックは同期レジスタを介して相互に接続できる。

アルゴリズムは、スレッドの最もクリティカルなパスに属するノードのサーチを開始する。また、アルゴリズムは、最大限の結合性を持つ２つのノードを近似性測定マトリクス（closeness measure matrix）から選択して、それらをいくつかの制約に依存する、同じ基本ブロック、同じマクロブロック、または異なるマクロブロックに配置する。

ここで、クラスタリング手順について説明する。

ノード群をどの階層にマップするかを決定するために、いくつかの条件、例えば次の諸条件について考える。

・Ｃ１：クリティカルパスに属する２つのノード。

・面積制約の検証：

Ｃ２１：基本ブロック面積。

Ｃ２２：マクロブロック面積。

Ｃ２３：全目的ＶＬＳＩ回路面積。

・Ｃ２：２つのノードの間の通信は、所定のしきい値を超えている。

その後で、ノードを次のようなブロックにまとめる。

・基本ブロック：

クリティカルパスに属している高く相互接続されたノード、または既にマップされている対で十分な余地がある場合に、

［C1 AND C21 AND C3］ＯＲ［NOT C1 AND NOT C21］

の条件で与えられる。

・マクロブロック：

クリティカルパスに属していない非常に相互接続されているノード、または十分な余地がない場合に

［NOT(C1) AND C3］ＯＲ［C1 AND NOT (C21)］

の条件で与えられる。

・異なるマクロブロック：

２つのノードの間の通信が所定のしきい値より小さく、再構成可能回路の面積または１つのマクロブロックのいずれもが全体のスレッドをサポートできない場合に、

［NOT (C4) AND (NOT (C22) OR NOT (C29))］

の条件で与えられる。

以上により、高く相互接続されたノードを局部的にまとめることにより配線の全長および配線の複雑さを最適に減ずることができる。

以上のことを、図１２ａ、図１２ｂ及び図１２ｃに示したスレッド１２０６を例にして、具体的に説明する。図１６から図１８に、図１２ａ、図１２ｂ及び図１２ｃに示したスレッド１２０６のノードクラスタリングの対応する結果を示す。まず、図１６に示すように、スレッド１２０６の全てのノードがラベルｖｉ（ｉ＝１～１９）で割り付けられ、クリティカルパス１６０２、１６０３が見出される。その後で、図１

８に示すようなクラスタツリー１６０５を構成するために、図１７に示すようなマトリクス１６０４が計算される。このツリーは、組合わせ操作の優先順位を表す。最も近似していることが見出されたノードｖ１７とｖ１８が、基本ブロック１６０７にマップすべき第１のペア候補を構成する。その最後の段階で、アルゴリズムが３つのマクロブロック１６０６と、６つの基本ブロック１６０７を生成する。

（ｂ）レジスタ・リタイミング（レジスタ再時間調整）

このタスクでは、待ち時間ｔｈがクロックサイクル周期ｔｃより長いスレッドに対して排他的に行われる。その場合、フロアー（ｘ）が、待ち時間がｔｃより長いスレッドのパスの数Ｃｐおよびｘに近い最小整数を表す場合に、挿入されるべきレジスタの数は１０である。そして、種々の組合わせを行うことによって、レジスタ・タイミングが最小面積を推定するためにとられる。

たとえば、ｔｃ＝６０ｎｓであると仮定し、図１９に示すような各ノードの遅延を考える。スレッドの待ち時間は、クリティカルパスの待ち時間であり、それは、

ｔｃ＝ｔｖ７＋ｔｖ７．６＋ｔｖ６＋ｔｖ３．６＋ｔｖ３＋ｔｖ３．１６＋ｔｖ１６，１７＋ｔｖ１７＋ｔｖ１７＋ｔｖ１７＋ｔｖ１７，ｖ１８＋ｔｖ１８＋ｔｖ１８，ｖ１９＋ｔｖ１９ｔｖ１９＝８６．５ｎｓ

に等しい。分割の数または挿入すべきレジスタの数は１０である。そして、レジスタ・リタイミングの３つの組合わせ１７０２、１７０３および１７０４が推定される。ここで、組合わせ１７０２は「ｖ１６・ｖ１７・ｖ１８」と「ｖ３・ｖ６・ｖ７」の組み合わせであり、組合わせ１７０３は「ｖ１６・ｖ１７」と「ｖ１８・ｖ３・ｖ６・ｖ７」の組み合わせであり、組合わせ１７０４は「ｖ１６・ｖ１７・ｖ１８・ｖ３」と「ｖ６・ｖ７」の組み合わせである。各組合わせ毎に、ライブラリ結合が、待ち時間制約に合致するように全てのＦＵに対して実行される。最小面積は、最適解として維持される。

（第３のスケジューリング）

スレッド処理の結果として、レジスタ・リタイミング段階中に、全体の面積を推定可能である。第３のスケジューリング５１０が、回路のスループットを徐々に増加す

るように行われる。これは、第２のスケジューリング５０８におけるアルゴリズムに類似するが、逆のやり方で優先度リストを用いる点でそれとは異なる。すなわち、この第３のスケジューリングでは、まず、最高優先度リスト（Ｐｌｉｓｔ１）から、類似性が低いスレッド対を選択する。そして、そのようなリストに属するスレッドが、その対応するグループから分離される。これと同様のプロセスが、全てのリストに対して繰り返される。

（第２の分割）

次に、コンテクストへのスレッドの分割（図５の５１１）が行われる。これは、ＤＲＬまたはマルチチップ回路に適用される。アルゴリズムは、シミュレートされたアニーリングをベースとしており、スレッドの間の結合性制約を最小にする包括的な解を提供する。２つのスレッドの間の結合性コストは、それらの間の共通変数の数であって、ＤＲＬの場合におけるコンテクストの間のデータを復元するために用いられるレジスタの数、またはマルチチップ回路の場合におけるオフ・チップ相互接続の数である。

以上の説明は、図１に示したシステム１０７のバック・エンド・コンパイラー１０５の処理を各処理部毎に説明したものである。図５に処理として示したブロックが、それぞれバック・エンド・コンパイラー１０５の各処理部に対応する。

以上説明したように、本発明によれば、プログラマに馴染みの深い高級記述言語による電子回路モデルの記述が可能であり、より正確なコスト見積もりを行うことができ、高品質の設計結果を提供することができる。

特許請求の範囲

1. 所望の電子回路モデルが所定の高級記述言語で記述された記述ファイルを構文解析して所定のグラフ構造を有する制御データ・フロー・グラフを生成する第１のステップと、

前記制御データ・フロー・グラフを、複数の連結されたノードの集合よりなる、特定の機能を果たすスレッドに分割し、該分割したスレッドを所定の面積制約および所定の待ち時間制約と合致するように最適化して、前記電子回路モデルに関する論理セルの数、機能、配置および配線の指定情報を得る第２のステップと

を含むことを特徴とするコンパイル方法。

2. 前記第２のステップにおける最適化が、機能ユニット、レジスタ、マルチプレクサのいずれかに関する面積と待ち時間との最低境界を推定することにより行われる請求項１に記載のコンパイル方法。

3. 前記第２のステップにおける最適化が、分割されたスレッドを所定の面積制約と合致するように最適化した後、さらにその最適化されたスレッドを所定の待ち時間制約と合致するように最適化することである請求項１に記載のコンパイル方法。

4. 前記第２のステップは、

前記所定の面積制約および待ち時間制約に基づく最適化を最上位の分割スレッドから順に行うトップ・ダウン処理ステップと、

前記トップ・ダウン・ステップにて最適化された下位の分割スレッドをいくつかのスレッドに分離して所定のコンテクストまたは所定の回路にまとめるダウン・トップ処理ステップとを含む請求項３に記載のコンパイル方法。

5. 前記トップ・ダウン処理ステップは、

前記制御データ・フロー・グラフを、複数の連結されたノードの集合よりなる、特定の機能を果たすスレッドに分割する第１の分割ステップと、

前記第１の分割ステップにて分割されたスレッドに対して所定の制御ステップおよび該ステップにおけるスレッドの移動レンジの割り付けを行うとともに、該制御ス

テップのおのおのに対して割り付けられたスレッドについて予め設定された複数の優先順位リストに従った優先順位を割り付ける第１のスケジューリング・ステップと、

前記第１のスケジューリング・ステップによる割り付けが施されたスレッドについてトータル面積を見積り、該トータル面積が所定の面積制約に合致するか否かを判定する第１の面積制約判定ステップと、

前記第１の面積制約判定ステップにて面積制約に合致しないと判定された場合に、前記第１の分割ステップにて分割されたスレッドの全てのスレッド対の組合わせについて面積に関する類似コストを算出する類似コスト算出ステップと、

前記類似コスト算出ステップにて算出された類似コストを参照して、異なる制御ステップに属し、かつ、より高い類似コストを有するスレッド対を前記スレッド対のうちから選択し、該選択したスレッド対を新たなスレッドとして他のスレッドとを組合わせて新たなスレッド対を得る第１のアロケーション・ステップと、

前記第１のアロケーション・ステップにて得られた新たなスレッド対についてトータル面積を見積って、該トータル面積が所定の面積制約に合致するか否かを判定する第２の面積制約判定ステップと、

前記第２の面積制約判定ステップにて面積制約に合致しないと判定された場合に、前記複数の優先順位リストに従って、優先順位の低いリストから順に、リストに含まれているスレッドについて、同じ制御ステップに属し、かつ、より高い類似コストを有するスレッド対を選択し、該選択したスレッド対を新たなスレッドとして他のスレッドと組合わせて新たなスレッド対を得るとともに、該新たなスレッド対が割り付けられた制御ステップを同じ内容の２つの制御ステップに細分化するアロケーションースケジューリング・ステップと、

前記第１または第２の面積制約判定ステップにて面積制約に合致した場合に、前記第１のアロケーション・ステップまたはアロケーションースケジューリング・ステップにて得られた新たなスレッド対について、前記面積制約と前記所定の待ち時間制約とのトレードオフを調べ、両制約に合致するように、ノードの配置および配線を行うスレッド処理ステップとを含み、

前記ダウン・トップ処理ステップは、

前記スレッド処理ステップにて配置配線されたスレッドについて、前記複数の優先順位リストに従って、優先順位の高いリストから順に、そのリストに含まれているスレッドのうちの類似性が低いスレッド対を選択して分離する第２のスケジューリング・ステップと、

前記第２のスケジューリング・ステップにて分離されたスレッド対を、そのスレッドの間の結合性制約が最小となるようなコンテクストまたは回路にまとめる第２の分割ステップとを含む請求項４に記載のコンパイル方法。

６．　前記スレッド処理ステップにおける所定の時間制約が、前記制御ステップにおけるスレッドの移動範囲を条件として規定した移動レンジ制約、前記制御ステップにおけるスレッドの時間的な重なり合いを条件として規定したスレッド共有制約、前記制御ステップを並列に実行するパイプライン処理の１つのループに属するスレッドの待ち時間を条件として規定したパイプライン制約の３つの制約である請求項５に記載のコンパイル方法。

７．　前記スレッド処理ステップは、

前記第１のアロケーション・ステップまたはアロケーション－スケジューリング・ステップにて得られた新たなスレッド対に、前記移動レンジ制約、スレッド共有制約、およびパイプライン制約の１つに合致しないスレッド対がある場合は、該スレッド対について、それら制約に合致する最小スレッド面積を持つ解を見出すスレッド調整ステップと、

前記スレッド調整ステップにて得られたスレッド対について、所定の結合性制約に基づいて遅延が最大となるクリティカルパスを調べてノードをクラスタにまとめ、所定のクロックサイクルより長い待ち時間を持つスレッドがある場合は、該スレッド中に挿入されるレジスタの数を求め、該レジスタのタイミングをとることで最小面積を推定し、前記所定の待ち時間制約に合致する解を見出すスレッド最適化ステップとを含む請求項５に記載のコンパイル方法。

８．　前記スレッド最適化ステップは、

前記スレッド調整ステップにて得られたスレッド対について、各スレッドのノードの近さを表わした近さマトリクスを計算するステップと;

前記近さマトリクスに基づいてノードをグループ化してノードクラスタツリーを作成するステップと、

前記クラスタツリーの各ノード対の結合性メトリクスに基づいて遅延が最大となるクリティカルパスを探索するステップと、

前記クラスタツリーの各ノード対を、前記クリティカルパスに属しているか否かでグループ化して基本ブロックとし、さらに該基本ブロックを最も近いもの同士でグループ化してマクロブロックとするステップとを含む請求項７に記載のコンパイル方法。

９. 前記スレッド調整ステップおよびスレッド最適化ステップにおける解を見出すステップが、対応する面積と遅延を有する所定のパラメータの設定が可能な機能ユニットの集合を供給するライブラリとの結合により行われる請求項７に記載のコンパイル方法。

１０. 第１の分割ステップにて分割されるスレッドは、連結ノード群の少なくとも１つの分岐の深さが所定のしきい値を超える場合に、同じＩＯポートを共有している２つの連続するメモリまたはＩＯアクセスの間に見出されるブロックとして、またはユーザーにより導入される明示状態機械として、あるいは前記制御データ・フロー・グラフの分岐結合ノードとして定められることを特徴とする請求項５に記載のコンパイル方法。

１１. 連続するＩＯアクセスの間に見出される前記スレッドは、前記制御ステップにメモリアクセスを含むループがある場合に、該ループの繰り返し中にメモリ並列が存在するかどうかを決定するループ拡張従属性が与えられることを特徴とする請求項１０に記載のコンパイル方法。

１２. 前記第２のステップにおける最適化に、面積および遅延の評価のためのレイアウトメトリクスを用いることを特徴とする請求項１に記載のコンパイル方法。

１３． 前記電子回路モデルが、所定数の基本素子で構成されるハードウェアセルよりなる請求項１に記載のコンパイル方法。

１４． 前記ハードウェアセルが、特定用途集積回路、フィールド・プログラマブル・ゲート・アレイ、または動的再構成可能論理のいずれかである請求項１３に記載のコンパイル方法。

１５． 所望の電子回路モデルが所定の高級記述言語で記述された記述ファイルを構文解析して所定のグラフ構造を有する制御データ・フロー・グラフを生成するフロント・エンド・コンパイラー手段と、

前記制御データ・フロー・グラフを、複数の連結されたノードの集合よりなる、特定の機能を果たすスレッドに分割し、該分割したスレッドを所定の面積制約および所定の待ち時間制約と合致するように最適化して、前記電子回路モデルに関する論理セルの数、機能、配置および配線の指定情報を得るバック・エンド・コンパイラー手段とを有することを特徴とする合成装置。

１６． 前記バック・エンド・コンパイラー手段は、機能ユニット、レジスタ、マルチプレクサのいずれかに関する面積と待ち時間との最低境界を推定することにより前記最適化を行うように構成されていることを特徴とする請求項１５に記載の合成装置。

１７． 前記バック・エンド・コンパイラー手段は、

前記制御データ・フロー・グラフを、複数の連結されたノードの集合よりなる、特定の機能を果たすスレッドに分割する第１の分割手段と、

前記第１の分割手段にて分割されたスレッドに対して所定の制御ステップおよび該ステップにおけるスレッドの移動レンジの割り付けを行うとともに、該制御ステップのおのおのに対して割り付けられたスレッドについて予め設定された複数の優先順位リストに従った優先順位を割り付ける第１のスケジューリング手段と、

前記第１のスケジューリング手段による割り付けが施されたスレッドについてトータル面積を見積り、該トータル面積が所定の面積制約に合致するか否かを判定する第１の面積制約判定手段と、

前記第１の面積制約判定手段にて面積制約に合致しないと判定された場合に、前記第１の分割手段にて分割されたスレッドの全てのスレッド対の組合わせについて面積に関する類似コストを算出する類似コスト算出手段と、

前記類似コスト算出手段にて算出された類似コストを参照して、異なる制御ステップに属し、かつ、より高い類似コストを有するスレッド対を前記スレッド対のうちから選択し、該選択したスレッド対を新たなスレッドとして他のスレッドとを組合わせて新たなスレッド対を得る第１のアロケーション手段と、

前記第１のアロケーション手段にて得られた新たなスレッド対についてトータル面積を見積って、該トータル面積が所定の面積制約に合致するか否かを判定する第２の面積制約判定手段と、

前記第２の面積制約判定手段にて面積制約に合致しないと判定された場合に、前記複数の優先順位リストに従って、優先順位の低いリストから順に、リストに含まれているスレッドについて、同じ制御ステップに属し、かつ、より高い類似コストを有するスレッド対を選択し、該選択したスレッド対を新たなスレッドとして他のスレッドと組合わせて新たなスレッド対を得るとともに、該新たなスレッド対が割り付けられた制御ステップを同じ内容の２つの制御ステップに細分化するアロケーションースケジューリング手段と、

前記第１または第２の面積制約判定手段にて面積制約に合致した場合に、前記第１のアロケーション手段またはアロケーションースケジューリング手段にて得られた新たなスレッド対について、前記面積制約と前記所定の待ち時間制約とのトレードオフを調べ、両制約に合致するように、ノードの配置および配線を行うスレッド処理手段と、

前記スレッド処理手段にて配置配線されたスレッドについて、前記複数の優先順位リストに従って、優先順位の高いリストから順に、そのリストに含まれているスレッドのうちの類似性が低いスレッド対を選択して分離する第２のスケジューリング手段と、

前記第２のスケジューリング手段にて分離されたスレッド対を、そのスレッドの間

の結合性制約が最小となるようなコンテクストまたは回路にまとめる第２の分割手段とを有することを特徴とする請求項１５に記載の合成装置。

１８．　前記所定の時間制約が、前記制御ステップにおけるスレッドの移動範囲を条件として規定した移動レンジ制約、前記制御ステップにおけるスレッドの時間的な重なり合いを条件として規定したスレッド共有制約、前記制御ステップを並列に実行するパイプライン処理の１つのループに属するスレッドの待ち時間を条件として規定したパイプライン制約の３つの制約である請求項１７に記載の合成装置。

１９．　前記スレッド処理手段は、所定の面積と遅延を有する所定のパラメータの設定が可能な機能ユニットの集合を供給するライブラリとの結合により、前記ノードの配置および配線を行うことを特徴とする請求項１７に記載の合成装置。

２０．　前記電子回路モデルが、所定数の基本素子で構成されるハードウェアセルよりなる請求項１５に記載の合成装置。

２１．　前記ハードウェアセルが、特定用途集積回路、フィールド・プログラマブル・ゲート・アレイ、または動的再構成可能論理のいずれかである請求項２０に記載の合成装置。

２２．　所望の電子回路モデルが所定の高級記述言語で記述された記述ファイルを構文解析して所定のグラフ構造を有する制御データ・フロー・グラフを生成する処理と、

前記制御データ・フロー・グラフを、複数の連結されたノードの集合よりなる、特定の機能を果たすスレッドに分割し、該分割したスレッドを所定の面積制約および所定の待ち時間制約と合致するように最適化して、前記電子回路モデルに関する論理セルの数、機能、配置および配線の指定情報を得る処理とをコンピュータに実行させるプログラムを記録した記録媒体。

開示の要約

所望の電子回路モデルが所定の高級記述言語で記述された記述ファイル１０２を構文解析して所定のグラフ構造を有する制御デ--タ・フロ--・グラフ１０４を生成するフロント・エンド・コンパイラー１０３と、制御データ・フロー・グラフ１０４を、複数の連結されたノードの集合よりなる、特定の機能を果たすスレッドに分割し、該分割したスレッドを所定の面積制約および所定の待ち時間制約と合致するように最適化して、上記電子回路モデルに関する論理セルの数、機能、配置および配線の指定情報を得るバック・エンド・コンパイラー１０５とを含む。本発明によるコンパイル方法によれば、プログラマに馴染みの深い高級記述言語による電子回路モデルの記述が可能で、より正確なコスト見積もりを行うことができる。